Computational Social Choice: Autumn 2010

Ulle Endriss
Institute for Logic, Language and Computation
University of Amsterdam

Plan for Today

Elections often have a *combinatorial structure:*

- Electing a committee of k members from amongst n candidates.
- During a referendum (in Switzerland, California, places like that), voters may be asked to vote on several propositions.

Clearly, the number of alternatives can quickly become *very large*. Can we somehow exploit the *internal structure* of the alternatives?

Today we will highlight some of the problems associated with voting in combinatorial domains and introduce some of the approaches that have been proposed to address these.

Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. Preference Handling in Combinatorial Domains: From AI to Social Choice. *AI Magazine*, 29(4):37–46, 2008.

The Paradox of Multiple Elections

Suppose 13 voters are asked to each vote *yes* or *no* on three issues; and we use the plurality rule for each issue independently to select a winning combination:

- 3 voters each vote for YNN, NYN, NNY.
- 1 voter each votes for YYY, YYN, YNY, NYY.
- No voter votes for NNN.

But then NNN wins: 7 out of 13 vote *no* on each issue.

This is an instance of the *paradox of multiple elections:* the winning combination receives the fewest number of votes.

S.J. Brams, D.M. Kilgour, and W.S. Zwicker. The Paradox of Multiple Elections. *Social Choice and Welfare*, 15(2):211–236, 1998.

The Problem

The problem of *voting in combinatorial domains:*

- Domain: variables $X_1, \ldots, X_p$ with finite domains $D_1, \ldots, D_p$
- Voters have preferences over set of combinations $D_1 \times \cdots \times D_p$.
- What should be the winning combination in $D_1 \times \cdots \times D_p$?

(1) Today we only consider *binary* variables. (2) Sometimes there are additional *constraints:* e.g., if we need to elect a committee of size k and each variable represents one candidate, then the number of yes-values in the winning combination has to be exactly k.

Next we sketch possible solutions that have been discussed in the literature. Our classification follows Chevaleyre et al. (2008), which in turn is based on work by Jérôme Lang.

Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. Preference Handling in Combinatorial Domains: From AI to Social Choice. *AI Magazine*, 29(4):37–46, 2008.

Possible Solutions

(1) Use your favourite voting procedure with the full set of combinations as the set of candidates.

(2) Do as for (1) but only require voters to rank their k most preferred combinations (e.g., for plurality we'd have $k = 1$).

(3) Select a small number of combinations and then use your favourite voting procedure to elect a candidate from amongst those.

(4) Ask voters to report their ballots using a compact representation language and apply your favourite voting procedure to the succinctly encoded ballots received ("combinatorial vote").

(5) Vote separately on each issue (as in the paradox), but —
   (a) identify conditions under which the paradox can be avoided;
   (b) find a novel way of aggregating the votes to select a winner;
   (c) do so sequentially rather than simultaneously.

Solution (1): Explicit Vote for Combinations

Idea: Vote for combinations directly: use your favourite voting procedure with the full set of combinations as the set of candidates.

Problem: This will only be possible in very small domains, certainly when the voting procedure requires a complete ranking of all the candidates (such as the Borda rule).

Example: Suppose there are six binary issues. This makes $2^6 = 64$ possible combinations. Hence, under the Borda rule, each voter has to choose between $64! \approx 1.27 \cdot 10^{89}$ possible ballots.

Solution (2): Vote for Top Combinations only

Idea: Vote for combinations directly, but only require voters to rank their k most preferred combinations, for some small number k.

Specifically, for the plurality rule $k = 1$ will suffice.

This clearly addresses the communication problems of Solution (1).

Problem: This may lead to almost random decisions, unless domains are fairly small and there are many voters.

Example: Suppose there are 10 binary issues to be decided upon and 100 voters. Then there are $2^{10} = 1024$ combinations to vote for. Under the plurality rule, chances are that no combination receives more than one vote (so the tie-breaking rule decides everything).

Solution (3): Vote for Selected Combinations only

Idea: Select a small number of combinations and then use your favourite voting procedure to elect a candidate from amongst those.

Problem: Who selects the candidate combinations? It is not at all clear what criteria should be used here. This gives the chooser (probably the election chair) undue powers and opens up new opportunities for controlling elections.

Solution (4): Combinatorial Vote

Idea: Ask voters to report their ballots using a compact preference *representation language* and apply your favourite voting procedure to the succinctly encoded ballots received.

Lang (2004) calls this approach *combinatorial vote*.

Discussion: This seems the most promising approach so far, although not too much is known to date what would be good choices for preference representation languages or voting procedures, or what algorithms to use to compute the winners. Also, complexity can be expected to be very high.

J. Lang. Logical Preference Representation and Combinatorial Vote. *Annals of Mathematics and Artificial Intelligence*, 42(1–3):37–71, 2004.

Preference Representation with Weighted Goals

Think of $\{X_1, \ldots, X_p\}$ as propositional variables and use propositional formulas to express *goals*. Give a numerical *weight* to each such goal.

- Cardinal preferences: a set G of weighted goals induces a *utility function* $u_G : D_1 \times \cdots \times D_p \to \mathbb{R}$, mapping each "model" M to

$$u_G(M) \quad = \quad \sum_{(\varphi,w)\in G[M]} w \quad \text{where } G[M] = \{(\varphi, w) \in G \mid M \models \varphi\}$$

- Ordinal preferences: Interpret weight as rank (lower is more important). Under the *lexicographic* form of aggregation, we prefer M to M' if there exists a k such that for all $j < k$ both M and M' satisfy the same number of goals of rank j, and M satisfies more goals of rank k.

Other forms of aggregation are possible (in both settings).

J. Lang. Logical Preference Representation and Combinatorial Vote. *Annals of Mathematics and Artificial Intelligence*, 42(1–3):37–71, 2004.

J. Uckelman. *More Than the Sum of its Parts: Compact Preference Representation over Combinatorial Domains*. PhD thesis, ILLC, University of Amsterdam, 2009.

Example

Use the language of *weighted goals* with *lexicographic aggregation* together with the *Borda rule*:

- Voter 1: $\{A{:}0, B{:}1\}$ induces order $AB \succ_1 A\bar{B} \succ_1 \bar{A}B \succ_1 \bar{A}\bar{B}$
- Voter 2: $\{A \vee \neg B{:}0\}$ induces order $A\bar{B} \sim_2 AB \sim_2 \bar{A}\bar{B} \succ_2 \bar{A}B$
- Voter 3: $\{\neg A{:}0, B{:}0\}$ induces order $\bar{A}B \succ_3 \bar{A}\bar{B} \sim_3 AB \succ_3 A\bar{B}$

As the induced orders need not be strict linear orders, we use a *generalisation of the Borda rule:* an alternative gets as many points as she dominates other alternatives. So we get these Borda counts:

$$AB : 3+1+1 = 5 \qquad \bar{A}B : 1+0+3 = 4$$
$$A\bar{B} : 2+1+0 = 3 \qquad \bar{A}\bar{B} : 0+1+1 = 2$$

So combination AB wins.

Combinatorial vote *proper* would be to compute the winner *directly* from the goal bases, without the detour via the induced orders.

Single Goals and Generalised Plurality

Next a couple of complexity results ... We will work with the following preference representation language and voting procedure:

- The *language of single goals* is an instance of the framework of weighted goals. Each voter specifies just one goal (an arbitrary propositional formula) with weight 1. We are only interested in the ordinal preference structure induced.
- Under the *generalised plurality rule*, a voter gives 1 point to each undominated alternative.

Here are two examples, for the set of variables $\{A, B\}$:

- The goal $\neg A \wedge B$ induces the order $\bar{A}B \succ AB \sim A\bar{B} \sim \bar{A}\bar{B}$, so only combination $\bar{A}B$ receives 1 point.
- The goal $A \vee B$ induces the order $AB \sim \bar{A}B \sim A\bar{B} \succ \bar{A}\bar{B}$, so combinations AB, $\bar{A}B$, $A\bar{B}$ receive 1 point each.

Winner Verification under Plurality

Define the following decision problem, for some preference representation language $\mathcal{L}$ and some voting procedure F:

AMONG-WINNERS($\mathcal{L}, F$)
Instance: Ballot profile $\underline{b}$ expressed in $\mathcal{L}$; candidate combination $x^\star$.
Question: Is $x^\star \in F(\underline{b})$?

The following result is due to Lang (2004):

Theorem 1 AMONG-WINNERS *is coNP-complete for the language of single goals and the generalised plurality rule.*

Recall that coNP is the complement of the complexity class NP and the complexity class of validity checking in propositional logic.

J. Lang. Logical Preference Representation and Combinatorial Vote. *Annals of Mathematics and Artificial Intelligence*, 42(1–3):37–71, 2004.

Proof

We show, equivalently to the claim, that checking whether $x^\star$ is *not* a winner under plurality is NP-complete:

- NP-membership: Let $\varphi_1, \ldots, \varphi_n$ be the goals of the voters. The plurality score of any candidate x can be computed by adding one point for each voter i with $x \models \varphi_i$ or φ_i being inconsistent (the two cases in which x is undominated). But to *compare* two candidates x and y, we only need to check $x \models \varphi_i$ and $y \models \varphi_i$, which we can do in polynomial time. Hence, if someone claims that $x^\star$ is *not* a winner and names a stronger candidate x, then this can be verified in polynomial time. ✓
- NP-hardness: By reduction from SAT. Let φ a formula for which we want to check satisfiability. Let p be a new propositional symbol; then φ is satisfiable iff $\varphi \wedge p$ is. Create a single voter with goal $\varphi \wedge p$. Pick candidate combination $x^\star$ with $x^\star \not\models p$ (must exist), i.e., also $x^\star \not\models \varphi \wedge p$. Then $x^\star$ is *not* a plurality winner iff there exists another candidate x with $x \models \varphi \wedge p$, i.e., iff $\varphi \wedge p$ is satisfiable. ✓

Verification of Condorcet Winners

Recall that a *Condorcet winner* is a candidate that beats every other candidate in pairwise majority contests.

CONDORCET-WINNER($\mathcal{L}$)
Instance: Ballot profile $\underline{b}$ expressed in $\mathcal{L}$; candidate combination $x^\star$.
Question: Is $x^\star$ a Condorcet winner?

We state this other result from Lang (2004) without proof:

Theorem 2 CONDORCET-WINNER *is coNP-complete for the language of single goals.*

Other results from the same paper show that the complexity of checking the winning status of candidates for more interesting preference languages tends to be above NP/coNP.

J. Lang. Logical Preference Representation and Combinatorial Vote. *Annals of Mathematics and Artificial Intelligence*, 42(1–3):37–71, 2004.

Solution (5a): Vote Separately Anyway

Idea: Try to identify conditions under which voting separately on each issue does not lead to paradoxes.

Discussion: To be precise, the "paradox" can never be avoided, but for some types of voter preferences this is not a worry. Recall that NNN won, even though it was nobody's favourite combination. If NNN is everyone's *least* favourite combination (which is possible in general), then this is really dramatic. But if voters only care about the individual decisions, then NNN is a reasonable outcome, as 9 out of 13 voters voted 2×N, for instance.

Partial solution: Voting on each issue separately is unproblematic if the voters all have *separable preferences* (i.e., if your preferences on X are always independent from how the other variables are instantiated). But separability is a pretty strong assumption.

Solution (5b): Vote Separately and Aggregate

Idea: Vote separately on each issue. However, don't just promote the issue-wise winners to the winning combination, but rather investigate other ways of aggregating the ballot information.

Discussion: Potentially interesting approach, *if* we can come up with a reasonable aggregation rule. We should choose a combination that somehow *minimises the distance* of the winning combination to the combinations the voters voted for (plurality-style ballots).

This raises two questions:

(1) What metric should we use to measure the distance between two combinations (e.g., winning combination and a ballot)?

(2) How should we aggregate the distances to individual ballots?

Minisum and Minimax Procedures

A possible answer to Question (1), for binary issues:

- A natural definition of the distance between two combinations is the *Hamming distance* (= number of issues where they differ).

Two possible possible answers to Question (2):

- *Minisum Procedure:* elect combinations that minimise the sum of distances to the combinations submitted as ballots.
- *Minimax Procedure:* elect combinations that minimise the maximal distance to any ballot. Proposed by Brams et al. (2007).

Observe that using the minisum procedure with the Hamming distance is just another way of circumscribing the standard approach: for each issue choose the value receiving the most support.

S.J. Brams, D.M. Kilgour, and M.R. Sanver. A Minimax Procedure for Electing Committees. *Public Choice*, 132:401–420, 2007.

Solution (5c): Sequential Voting

Idea: Vote separately on each issue, but do so sequentially to give voters the opportunity to make their vote for one issue dependent on other issues already decided.

We will discuss two results for this approach:

- A simple result showing that sequential voting does address some of the problems raised by the multiple election paradox.
- A stronger result in case we can make the assumption that voter preferences are induced by "compatible" CP-nets (another compact preference representation language).

Sequential Voting and Condorcet Losers

Recall that a *Condorcet loser* is a candidate that loses against any other candidate in a pairwise contest. Electing a CL is very bad.

Example: NNN (the winning combination) in the original multiple election paradox is such a Condorcet loser.

Lacy and Niou (2000) show that sequential voting can avoid this:

Theorem 3 *Sequential voting (with plurality) over binary issues never results in a winning combination that is a Condorcet loser.*

Proof: Just think what happens during the election for the final issue. The winning combination cannot be a Condorcet loser, because it does, at least, win against the other combination that was still possible after the penultimate election. ✓

D. Lacy and E.M.S. Niou. A Problem with Referendums. *Journal of Theoretical Politics*, 12(1):5–31, 2000.

CP-Nets

A CP-net over a set of variables $\{X_1, \ldots, X_p\}$ consists of

- a *directed graph* G over $\{X_1, \ldots, X_p\}$ and
- a *conditional preference table* (CPT) for each X_i, fixing a total order on values of X_i for each instantiation of X_i's parents in G.

This induces a (partial) preference order:

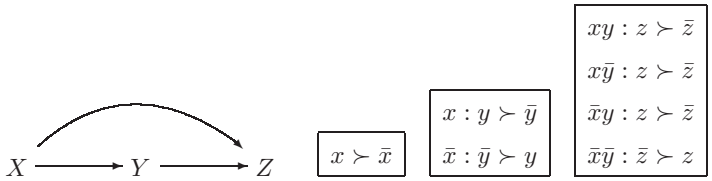
(1) If $\vec{y}$ is an instantiation of the parents of X, and $\vec{y} : x \succ x'$ is part of the CPT for X, then prefer x to x' given $\vec{y}$, *ceteris paribus*, i.e., prefer $x\vec{y}\vec{z}$ to $x'\vec{y}\vec{z}$ for any instantiation $\vec{z}$ of all other variables.

(2) Take the transitive closure of the above.

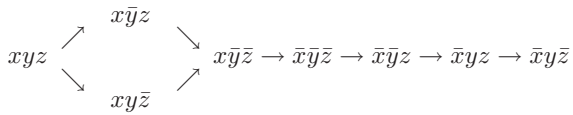
C. Boutilier, R.I. Brafman, C. Domshlak, H.H. Hoos, and D. Poole. CP-nets: A Tool for Representing and Reasoning with Conditional *Ceteris Paribus* Preference Statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.

Example

Consider the following CP-net, consisting of a graph on $\{X, Y, Z\}$ and conditional preference tables for X, Y and Z:



This CP-net induces the following (partial) preference order:



Sequential Voting with CP-Nets

Now suppose that every voter's preferences can be modelled as an (acyclic) CP-net and that there is an ordering of the issues $X_1, \ldots, X_p$ that is compatible with each of the underlying graphs.

Idea: Vote sequentially in this order!

Lang and Xia (2009) have shown (proof omitted but easy):

Theorem 4 *The rule of sequential voting with CP-nets satisfies the Condorcet principle whenever all of the local voting procedures do.*

This is useful, particularly when the issues are binary (as then any reasonable local procedure will be Condorcet-consistent).

But note that each voter's graph must be compatible with the *same agenda* (the order of issues to vote on). So this is still quite restrictive.

J. Lang and L. Xia. Sequential Composition of Voting Rules in Multi-issue Domains. *Mathematical Social Sciences*, 57(3):304–324, 2009.

Other Properties

Lang and Xia (2009) also investigate for several other properties of voting rules (besides Condorcet-consistency) whether they *transfer* from the local voting rules to the global rule under sequential voting with CP-nets. Three examples:

- *Anonymity* does transfer.
- *Neutrality* does *not* transfer.
- *Reinforcement* also does transfer.

J. Lang and L. Xia. Sequential Composition of Voting Rules in Multi-issue Domains. *Mathematical Social Sciences*, 57(3):304–324, 2009.

Summary

We have seen several approaches for tackling the problem of voting in combinatorial domains (i.e., for voting in multi-issue elections). To date, no clear solution has emerged.

Two approaches seem promising (and require careful analysis):

- Use compact representation languages to encode ballots and directly operate on these.

  Issues: What languages? What voting procedures? Algorithms? Complexity? How deal with incompletely specified preferences?

- Vote on separate issues separately, or possibly on subsets of issues at a time, simultaneously or sequentially.

  Issues: When is this safe? How do you package issues? In which order do you hold local elections? How do you aggregate the outcomes of local elections? Which properties transfer from local elections to the global election, which don't?

What next?

The next lecture will deal with informational and communicational issues in voting, and address questions such as:

- How much information do we need to elicit from the electorate to be able to compute the winners of an election?
- What can we say about the status of an election when we only have incomplete information regarding preferences/ballots?